TreeCollections Class

1. insertT(String name, String capital, int population)

```
Create Node newNode passing the parameters into it's constructor
IF(root == null)
        root = newNode
ELSE
        create Node current and assign it root's value
        create Node parent
        WHILE(true)
                parent equals current
                IF(newNode's country name is not the same as current's country name)
                        current is now current's left child
                        IF(current is null)
                                parent's left child is newNode
                                return
                ELSE
                        current is now current's right child
                        IF(current is null)
                                parent's right child is newNode
                                return
```

2. lnr(Node localRoot)

```
IF(local root is not null)
        recursively call lnr(pass localRoot's left child)
        output formated text
        recursively call lnr(pass localRoot's right child)
```

3. rnl(Node localRoot)

```
IF(local root is not null)
        recursively call rnl(pass localRoot's right child)
        output formated text
        recursively call rnl(pass localRoot's left child)
```

4. lnrIterative()

```
Create a Stack and call it lnrIt with an array size of 21
Create node current and assign it root's value
WHILE(true)
        IF(current is not null)
                push current onto the lnrIt stack
                current is now current's left child
        ELSE
                IF(lnrIt is empty)
                        return
                current is now what lnrIt pops
```

```
        output formatted text
        current is now current's right child
```

5. rnlIterative()

```
Create a Stack and call it lnrIt with an array size of 21
Create node current and assign it root's value
WHILE(true)
    IF(current is not null)
        push current onto the lnrIt stack
        current is now current's right child
    ELSE
        IF(lnrIt is empty)
            return
        current is now what lnrIt pops
        output formatted text
        current is now current's left child
```

6. getSuccessor(Node delNode)

```
Create Node successorParent and assign it to delNode's values
Create Node successor and assign it to delNode's values
Create Node current and assign it to delNode's right child
WHILE(current is not null)
    successorParent is assigned successor's values
    successor is assigned current's values
    current is assigned current's left child
IF(successor is not delNode's right child)
    successorParent's left child is now successor's right child
    successor right child is now assigned delNode's right child
RETURN successor node
```

7. delete(String key)

```
Create Node current and assign it root's values
Create parent and assign it root's values
Create boolean variable isLeftChild and assign it true
WHILE(current's country name is not the search key)
    parent is assigned current's values
    IF(key is not equal to current's country name)
        isLeftChild is true
        current is assigned current's left child values
    ELSE
        isLeftChild is false
        current is assigned current's right child values
    IF(current is null)
        RETURN false
```

```
IF(current's left and right children are null)
        IF(current is root)
                root is null
        ELSE IF(isLeftChild)
                parent's left child is null
        ELSE
                parent's right child is null
ELSE IF(current's right child is null)
         IF(current is root)
                root is now current's left child
        ELSE IF(isLeftChild)
                parent's left child is now current's left child
        ELSE
                parent's right child is now current's left child
ELSE IF(current's left child is null)
        IF(current is root)
                root is now current's right child
        ELSE IF
                parent's left child is now current's right child
        ELSE
                parent's right child is now current's right child
ELSE
        Create Node successor and assign it the value of getSuccessor(current)
        IF(current is root)
                root is assigned successor's values
        ELSE IF(isLeftChild)
                parent's left child is assigned successor's values
        ELSE
                parent' right child is assigned successor's values
        successor's left child is assigned current's left child
RETURN true
```